Domain Save Image Format

David Vrabel <david.vrabel@citrix.com>

Draft B

Contents

1 Introduction

1.1 Revision History

Version	Date	Changes
Draft A	6 Feb 2014	Initial draft.
Draft B	10 Feb 2014	Corrected image header field widths.
		Minor updates and clarifications.

1.2 Purpose

The *domain save image* is the context of a running domain used for snapshots of a domain or for transferring domains between hosts during migration.

There are a number of problems with the format of the domain save image used in Xen 4.4 and earlier (the *legacy format*).

- Dependant on toolstack word size. A number of fields within the image are native types such as `unsigned long` which have different sizes between 32-bit and 64-bit hosts. This prevents domains from being migrated between 32-bit and 64-bit hosts.
- There is no header identifying the image.
- The image has no version information.

A new format that addresses the above is required.

ARM does not yet have have a domain save image format specified and the format described in this specification should be suitable.

2 Overview

The image format consists of two main sections:

- *Headers*
- *Records*

2.1 Headers

There are two headers: the *image header*, and the *domain header*. The image header describes the format of the image (version etc.). The *domain header* contains general information about the domain (architecture, type etc.).

2.2 Records

The main part of the format is a sequence of different *records*. Each record type contains information about the domain context. At a minimum there is a END record marking the end of the records section.

2.3 Fields

All the fields within the headers and records have a fixed width.

Fields are always aligned to their size.

Padding and reserved fields are set to zero on save and must be ignored during restore.

Integer (numeric) fields in the image header are always in big-endian byte order.

Integer fields in the domain header and in the records are in the endianess described in the image header (which will typically be the native ordering).

3 Headers

3.1 Image Header

The image header identifies an image as a Xen domain save image. It includes the version of this specification that the image complies with.

Tools supporting version V of the specification shall always save images using version V. Tools shall support restoring from version V and version V - 1. Tools may additionally support restoring from earlier versions.

The marker field can be used to distinguish between legacy images and those corresponding to this specification. Legacy images will have at one or more zero bits within the first 8 octets of the image.

Fields within the image header are always in *big-endian* byte order, regardless of the setting of the endianness bit.

```
 0     1     2     3     4     5     6     7 octet
+-------------------------------------------------+
| marker                                          |
+-----------------------+-------------------------+
| id                    | version                 |
+-----------+-----------+-------------------------+
| options   |                                     |
+-----------+-------------------------------------+
```

Field	Description
marker	0xFFFFFFFFFFFFFFFF.
id	0x58454E46 (“XENF” in ASCII).
version	0x00000001. The version of this specification.
options	bit 0: Endianness. 0 = little-endian, 1 = big-endian.
	bit 1–15: Reserved.

3.2 Domain Header

The domain header includes general properties of the domain.

```
 0      1     2     3     4     5     6     7 octet
+-----------+-----------+-----------+-------------+
| arch      | type      | page_shift| (reserved)  |
+-----------+-----------+-----------+-------------+
```

Field	Description
arch	0x0000: Reserved.
	0x0001: x86.
	0x0002: ARM.
type	0x0000: Reserved.
	0x0001: x86 PV.
	0x0002 - 0xFFFF: Reserved.
page_shift	Size of a guest page as a power of two.
	i.e., page size = $2^{\text{page\_shift}}$.

4 Records

A record has a record header, type specific data and a trailing footer. If body_length is not a multiple of 8, the body is padded with zeroes to align the checksum field on an 8 octet boundary.

```
 0     1     2     3     4     5     6     7 octet
+-----------------------+-------------------------+
| type                  | body_length             |
+-----------+-----------+-------------------------+
| options   | (reserved)                          |
+-----------+-------------------------------------+
...
Record body of length body_length octets followed by
0 to 7 octets of padding.
...
+-----------------------+-------------------------+
| checksum              | (reserved)              |
+-----------------------+-------------------------+
```

Field	Description
type	0x00000000: END
	0x00000001: PAGE_DATA
	0x00000002: VCPU_INFO
	0x00000003: VCPU_CONTEXT
	0x00000004: X86_PV_INFO
	0x00000005: P2M
	0x00000006 - 0xFFFFFFFF: Reserved
body_length	Length in octets of the record body.
options	Bit 0: 0 - checksum invalid, 1 = checksum valid.
	Bit 1–15: Reserved.
checksum	CRC–32 checksum of the record body (including any trailing padding), or 0x00000000 if the checksum field is invalid.

The following sub-sections specify the record body format for each of the record types.

4.1 END

A end record marks the end of the image.

```
 0     1     2     3     4     5     6     7 octet
+-------------------------------------------------+
```

The end record contains no fields; its body_length is 0.

4.2 PAGE_DATA

The bulk of an image consists of many PAGE_DATA records containing the memory contents.

```
 0     1     2     3     4     5     6     7 octet
+------------------------+-------------------------+
| count (C)              | (reserved)              |
+------------------------+-------------------------+
| pfn[0]                                           |
+--------------------------------------------------+
...
+--------------------------------------------------+
| pfn[C-1]                                         |
+--------------------------------------------------+
| page_data[0]...                                  |
...
+--------------------------------------------------+
| page_data[N-1]...                                |
...
+--------------------------------------------------+
```

Field	Description
count	Number of pages described in this record.
pfn	An array of count PFNs. Bits 63–60 contain the XEN_DOMCTL_PFINFO_* value for that PFN.
page_data	page_size octets of uncompressed page contents for each page set as present in the pfn array.

4.3 VCPU_INFO

> [This is a combination of parts of the extended-info and XC_SAVE_ID_VCPU_INFO chunks.]

The VCPU_INFO record includes the maximum possible VCPU ID. This will be followed a VCPU_CONTEXT record for each online VCPU.

```
 0     1     2     3     4     5     6     7 octet
+------------------------+------------------------+
| max_vcpu_id            | (reserved)             |
+------------------------+------------------------+
```

Field	Description
max_vcpu_id	Maximum possible VCPU ID.

4.4 VCPU_CONTEXT

The context for a single VCPU.

```
 0     1     2     3     4     5     6     7 octet
+-----------------------+-------------------------+
| vcpu_id               | (reserved)              |
+-----------------------+-------------------------+
| vcpu_ctx...                                     |
...
+-------------------------------------------------+
```

Field	Description
vcpu_id	The VCPU ID.
vcpu_ctx	Context for this VCPU.

[vcpu_ctx format TBD.]

4.5 X86_PV_INFO

[This record replaces part of the extended-info chunk.]

```
 0     1     2     3     4     5     6     7 octet
+-----+-----+-----+-------------------------------+
| w   | ptl | o   | (reserved)                    |
+-----+-----+-----+-------------------------------+
```

Field	Description
guest_width (w)	Guest width in octets (either 4 or 8).
pt_levels (ptl)	Number of page table levels (either 3 or 4).
options (o)	Bit 0: 0 - no VMASST_pae_extended_cr3, 1 - VMASST_pae_extended_cr3. Bit 1–7: Reserved.

4.6 P2M

[This is a more flexible replacement for the old p2m_size field and p2m array.]

The P2M record contains a portion of the source domain's P2M. Multiple P2M records may be sent if the source P2M changes during the stream.

```
 0     1     2     3     4     5     6     7 octet
+-------------------------------------------------+
| pfn_begin                                       |
+-------------------------------------------------+
| pfn_end                                         |
+-------------------------------------------------+
| mfn[0]                                          |
+-------------------------------------------------+
...
+-------------------------------------------------+
| mfn[N-1]                                        |
+-------------------------------------------------+
```

Field	Description
pfn_begin	The first PFN in this portion of the P2M
pfn_end	One past the last PFN in this portion of the P2M.
mfn	Array of (pfn_end - pfn-begin) MFNs corresponding to the set of PFNs in the range [pfn_begin, pfn_end).

5 Layout

The set of valid records depends on the guest architecture and type.

5.1 x86 PV Guest

An x86 PV guest image will have in this order:

1. Image header
2. Domain header
3. X86_PV_INFO record
4. At least one P2M record
5. At least one PAGE_DATA record
6. VCPU_INFO record
7. At least one VCPU_CONTEXT record
8. END record

6 Legacy Images (x86 only)

Restoring legacy images from older tools shall be handled by translating the legacy format image into this new format.

It shall not be possible to save in the legacy format.

There are two different legacy images depending on whether they were generated by a 32-bit or a 64-bit toolstack. These shall be distinguished by inspecting octets 4–7 in the image. If these are zero then it is a 64-bit image.

Toolstack	Field	Value
64-bit	Bit 31–63 of the p2m_size field	0 (since p2m_size $< 2^{32}$)
32-bit	extended-info chunk ID (PV)	0xFFFFFFFF
32-bit	Chunk type (HVM)	< 0
32-bit	Page count (HVM)	> 0

Table 1: Possible values for octet 4–7 in legacy images

This assumes the presence of the extended-info chunk which was introduced in Xen 3.0.

7 Future Extensions

All changes to this format require the image version to be increased.

The format may be extended by adding additional record types.

Extending an existing record type must be done by adding a new record type. This allows old images with the old record to still be restored.

The image header may be extended by *appending* additional fields. In particular, the `marker`, `id` and `version` fields must never change size or location.